# Project 1

Mike Keating, Hayden Morgan

## Project 1

### Setting Things Up

### Creating the Repo

- GitHub repo created by Mike
- RStudio project created
- Hayden added as a collaborator and membership accepted
- Format set to PDF

### Collaboration Workflow

- Task distribution and timeline established
- Decided to each work on own branches

### .qmd Format

All messages and warnings that come from librarying packages should be turned off using the appropriate code chunk option.

```r
library("tidyverse")
library("ggplot2")
```

## First Steps

### Question 1: Selecting Columns

Read in one section of the data. This data is available at https://www4.stat.ncsu.edu/~online/datasets/EDU01a.csv.

Select only the following columns:

- Area_name (rename area_name)
- STCOU
- Any column that ends in "D"

```
#NOTE: EDU01a, EDU01b, divisions, and Mastdata files are all in the project folder.

edu01a <- read_csv("data/EDU01a.csv",
                   col_select = c(Area_name, STCOU, ends_with("D")),
                    show_col_types = FALSE) |>
  rename(
    area_name = Area_name
  )
```

Display the first 5 rows of your new data set to show that you created this correctly. Note: Do not save over your new data set with just the first 5 rows, simply just show the first 5 rows.

```
head(edu01a, 5)
```

```
# A tibble: 5 x 12
  area_name     STCOU EDU010187D EDU010188D EDU010189D EDU010190D EDU010191D
  <chr>         <chr>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 UNITED STATES 00000   40024299   39967624   40317775   40737600   41385442
2 ALABAMA       01000     733735     728234     730048     728252     725541
3 Autauga, AL   01001       6829       6900       6920       6847       7008
4 Baldwin, AL   01003      16417      16465      16799      17054      17479
5 Barbour, AL   01005       5071       5098       5068       5156       5173
# i 5 more variables: EDU010192D <dbl>, EDU010193D <dbl>, EDU010194D <dbl>,
#   EDU010195D <dbl>, EDU010196D <dbl>
```

## Question 2: Converting to Long Format

Convert the data into long format where each row has only one enrollment value for that Area_name. Display the first 5 rows of your new data set to show that you created this correctly.

```
edu01a_long <- edu01a |>
                  pivot_longer(cols = 3:12, #retain area_name + STCOU
                               names_to = "EDU_D", #named after unique
                               # ending "D" per Q1
                               values_to = "Enrollment")

#Displaying first 5 rows below
head(edu01a_long, 5)
```

```
# A tibble: 5 x 4
  area_name     STCOU EDU_D      Enrollment
  <chr>         <chr> <chr>           <dbl>
1 UNITED STATES 00000 EDU010187D   40024299
2 UNITED STATES 00000 EDU010188D   39967624
3 UNITED STATES 00000 EDU010189D   40317775
4 UNITED STATES 00000 EDU010190D   40737600
5 UNITED STATES 00000 EDU010191D   41385442
```

## Question 3: Assign Year and State

One of the new columns should now correspond to the old column names that end with a "D". All columns in these census data files will have this similar format. The first three characters represent the survey with the next four representing the type of value you have from that survey. The last two digits prior to the "D" represent the year of the measurement. For more about the variables see the data information sheet Mastdata.xls).

- Parse the string to pull out the year and convert the year into a numeric value such as 1997 or 2002.
- Grab the first three characters and following four digits to create a new variable representing which measurement was grabbed.
- Hint: Check out the substr() function from base r

```
# Parse the string to pull out year
# It looks like every year is pre-2000, but lets plan for up to 2025
# This assumes there is no data from 1925 or earlier
# Treating year as numeric for now
long_updated <- edu01a_long |>
  mutate(year = as.numeric(substr(EDU_D, 8,9)),
         measurement = substr(EDU_D, 1,7)) |>
  mutate(year = ifelse(year < 26, year + 2000, year + 1900))

head(long_updated)
```

```
# A tibble: 6 x 6
  area_name     STCOU EDU_D      Enrollment  year measurement
  <chr>         <chr> <chr>           <dbl> <dbl> <chr>
1 UNITED STATES 00000 EDU010187D   40024299  1987 EDU0101
2 UNITED STATES 00000 EDU010188D   39967624  1988 EDU0101
3 UNITED STATES 00000 EDU010189D   40317775  1989 EDU0101
4 UNITED STATES 00000 EDU010190D   40737600  1990 EDU0101
5 UNITED STATES 00000 EDU010191D   41385442  1991 EDU0101
6 UNITED STATES 00000 EDU010192D   42088151  1992 EDU0101
```

## Question 4: Split County and Non-County

Create two data sets

- one data set that contains only non-county data
- one data set that contains only county level data

Note that all county measurements have the format "County Name, DD" where "DD" represents the state. This can be used to subset the data. For the county level data, add a class to the tibble called county. Similarly, add a class to the non-county data called state.

```
#For county tibble
county_match <- grep(pattern = ", \\w\\w", long_updated$area_name)
county_tibble <- long_updated[county_match,]
class(county_tibble) <- c("county", class(county_tibble))

#For state tibble
state_match <- grep(pattern = ", \\w\\w", long_updated$area_name, invert = T)
state_tibble <- long_updated[state_match,]
class(state_tibble) <- c("state", class(state_tibble))
```

Print the first 10 rows of each tibble by including county_tibble and state_tibble in your code chunk.

```
head(county_tibble, 10)
```

```
# A tibble: 10 x 6
   area_name   STCOU EDU_D      Enrollment  year measurement
   <chr>       <chr> <chr>           <dbl> <dbl> <chr>
 1 Autauga, AL 01001 EDU010187D       6829  1987 EDU0101
 2 Autauga, AL 01001 EDU010188D       6900  1988 EDU0101
 3 Autauga, AL 01001 EDU010189D       6920  1989 EDU0101
 4 Autauga, AL 01001 EDU010190D       6847  1990 EDU0101
 5 Autauga, AL 01001 EDU010191D       7008  1991 EDU0101
 6 Autauga, AL 01001 EDU010192D       7137  1992 EDU0101
 7 Autauga, AL 01001 EDU010193D       7152  1993 EDU0101
 8 Autauga, AL 01001 EDU010194D       7381  1994 EDU0101
 9 Autauga, AL 01001 EDU010195D       7568  1995 EDU0101
10 Autauga, AL 01001 EDU010196D       7834  1996 EDU0101
```

```
head(state_tibble, 10)
```

```
# A tibble: 10 x 6
   area_name     STCOU EDU_D      Enrollment  year measurement
   <chr>         <chr> <chr>           <dbl> <dbl> <chr>
 1 UNITED STATES 00000 EDU010187D   40024299  1987 EDU0101
 2 UNITED STATES 00000 EDU010188D   39967624  1988 EDU0101
 3 UNITED STATES 00000 EDU010189D   40317775  1989 EDU0101
 4 UNITED STATES 00000 EDU010190D   40737600  1990 EDU0101
 5 UNITED STATES 00000 EDU010191D   41385442  1991 EDU0101
 6 UNITED STATES 00000 EDU010192D   42088151  1992 EDU0101
 7 UNITED STATES 00000 EDU010193D   42724710  1993 EDU0101
 8 UNITED STATES 00000 EDU010194D   43369917  1994 EDU0101
 9 UNITED STATES 00000 EDU010195D   43993459  1995 EDU0101
10 UNITED STATES 00000 EDU010196D   44715737  1996 EDU0101
```

## Question 5: Assign State to County Tibble

For the county level tibble, create a new variable that describes which state one of these county measurements corresponds to (the two digit abbreviation is fine, see substr()).

```
#I prefer to split the string based on delimiter (comma) instead of indexing
#Example
string <- "Autauga, AL"
split <- str_split(string, ",", simplify = TRUE)[,-1] # We return a chr matrix,
# and we only care about the last (second) entry

print(split)
```

```
[1] " AL"
```

```
print("Removing space")
```

```
[1] "Removing space"
```

```
clean_split <- str_trim(split)
print(clean_split)
```

```
[1] "AL"
```

```
#Create state variable for county tibble

county_tibble <- county_tibble |>
  mutate(state = str_trim(str_split(area_name, ",", simplify = TRUE)[,-1]))

county_tibble #to show that the addition of the variable was successful
```

```
# A tibble: 31,450 x 7
   area_name   STCOU EDU_D       Enrollment  year measurement state
   <chr>       <chr> <chr>            <dbl> <dbl> <chr>       <chr>
 1 Autauga, AL 01001 EDU010187D        6829  1987 EDU0101     AL
 2 Autauga, AL 01001 EDU010188D        6900  1988 EDU0101     AL
 3 Autauga, AL 01001 EDU010189D        6920  1989 EDU0101     AL
 4 Autauga, AL 01001 EDU010190D        6847  1990 EDU0101     AL
 5 Autauga, AL 01001 EDU010191D        7008  1991 EDU0101     AL
 6 Autauga, AL 01001 EDU010192D        7137  1992 EDU0101     AL
 7 Autauga, AL 01001 EDU010193D        7152  1993 EDU0101     AL
 8 Autauga, AL 01001 EDU010194D        7381  1994 EDU0101     AL
 9 Autauga, AL 01001 EDU010195D        7568  1995 EDU0101     AL
10 Autauga, AL 01001 EDU010196D        7834  1996 EDU0101     AL
# i 31,440 more rows
```

### Question 6: Assign Division to State Tibble

For the non-county level tibble, create a new variable called “division” corresponding to the state’s classification of division here. If row corresponds to a non-state (i.e. UNITED STATES), return ERROR for the division. Hint: Use %in% and consider if_else or case_when logic.

Instead of writing ifelse statements manually for every division, we are going to instead read the divisions straight from Wikipedia and assign the correct division to any given state.

We can scrape a Wikipedia table using the rvest package.

Source: StackOverflow

```
library(rvest) # rvest is in the tidyverse package
```

```
# Since we don't want to always have to connect to the url to read our data,
# let's check if we have already saved it
if (file.exists("data/divisions.csv")){
  print("Division data already downloaded from Wikipedia")
  print("Reading .csv file")
  divisions <- read_csv("data/divisions.csv", show_col_types = FALSE)
} else {
  print("No division data found. Downloading from Wikipedia...")
  wiki <- read_html(x =
"https://en.wikipedia.org/wiki/List_of_regions_of_the_United_States",
package="xml2")
  wiki |> html_elements(".wikitable") |> html_table() -> wiki_tables
  # There is only one table, so the first one will give us what we want
  divisions <- wiki_tables[1]
  # Write the file to csv
  write.csv(divisions, file= "data/divisions.csv")
  print("data/divisions.csv successfully created!")
}
```

```
[1] "Division data already downloaded from Wikipedia"
[1] "Reading .csv file"
```

```
divisions
```

```
# A tibble: 9 x 4
   ...1 Region    Division           States
  <dbl> <chr>     <chr>              <chr>
1     1 Northeast New England        Connecticut Maine Massachusetts New Hampsh~
```

```
2     2 Northeast Mid-Atlantic       New Jersey New York Pennsylvania
3     3 Midwest   East North Central Illinois Indiana Michigan Ohio Wisconsin
4     4 Midwest   West North Central Iowa Kansas Minnesota Missouri Nebraska No~
5     5 South     South Atlantic     Delaware District of Columbia Florida Geor~
6     6 South     East South Central Alabama Kentucky Mississippi Tennessee
7     7 South     West South Central Arkansas Louisiana Oklahoma Texas
8     8 West      Mountain           Arizona Colorado Idaho Montana Nevada New ~
9     9 West      Pacific            Alaska California Hawaii Oregon Washington
```

Note how all states in any given region are stored in the same cell, separated by spaces. We can either transform the States column by splitting up the states or leave as is and process the state correctly when reading our other datasets.

We can filter columns by the state in our divisions tibble by using if_any and str_detect.

```r
# Make uppercase to improve matching
divisions$States <-divisions$States |> toupper()

get_division_for_state <- function(state_name){

  # Check for the state name in the divisions df and filter
  # Assumes state only appears once in the tibble
  # Add word boundaries to our regex to avoid substring matching
  # E.g "Kansas" shouldn't match "Arkansas"
  match_pattern <- paste0("\\b", toupper(state_name), "\\b")
  division_row <- divisions |>
    filter(if_any(States, ~str_detect(.x, match_pattern)))
  division <- division_row$Division
  # Return "ERROR" if there is no match to state
  if (length(division) == 0){
    return ("ERROR")
  }
  else {
    return (division)
  }

}
```

```r
# Apply our function to the non county tibble

state_tibble_test <- state_tibble |> mutate(division =
                     map_chr(area_name, get_division_for_state))
tail(state_tibble_test)
```

```
# A tibble: 6 x 7
  area_name STCOU EDU_D      Enrollment  year measurement division
  <chr>     <chr> <chr>           <dbl> <dbl> <chr>       <chr>   
1 WYOMING   56000 EDU010191D      98782  1991 EDU0101     Mountain
2 WYOMING   56000 EDU010192D     101715  1992 EDU0101     Mountain
3 WYOMING   56000 EDU010193D     100729  1993 EDU0101     Mountain
4 WYOMING   56000 EDU010194D     100899  1994 EDU0101     Mountain
5 WYOMING   56000 EDU010195D     100369  1995 EDU0101     Mountain
6 WYOMING   56000 EDU010196D      99859  1996 EDU0101     Mountain
```

## Function Wrapping

### Function 1: Step 1 & Step 2

Write one function that combines Steps 1 and 2 above. Give an optional argument (that is it has a default value) that allows the user to specify the name of the column representing the value (enrollment for these data sets).

```r
select_and_convert <- function(data_path_in_quotes,
                               value_colname = "Enrollment"){
  edu <- read_csv(data_path_in_quotes,
                  col_select = c(Area_name, STCOU, ends_with("D")),
                  show_col_types = FALSE) |>
  rename(
    area_name = Area_name
  )

  edu_long <- edu |>
                pivot_longer(cols = 3:12,
                             names_to = "EDU_D",
                             values_to = value_colname)
  print(head(edu_long, 5))
  return(edu_long)
}

#making sure the function works
function1 <- select_and_convert("data/EDU01b.csv")
```

```
# A tibble: 5 x 4
  area_name     STCOU EDU_D      Enrollment
  <chr>         <chr> <chr>           <dbl>
```

```
1 UNITED STATES 00000 EDU010197D   44534459
2 UNITED STATES 00000 EDU010198D   46245814
3 UNITED STATES 00000 EDU010199D   46368903
4 UNITED STATES 00000 EDU010200D   46818690
5 UNITED STATES 00000 EDU010201D   47127066
```

### Function 2: Step 3

Write a function that takes the output from Step 2 and performs Step 3

```
get_year_and_measurement <-function(long_data){
  print("Updating long data with year and measurement")
  long_data_updated <- long_data |>
    mutate(year = as.numeric(substr(EDU_D, 8,9)),
           measurement = substr(EDU_D, 1,7)) |>
    mutate(year = ifelse(year < 26, year + 2000, year + 1900))

  return (long_data_updated)
}

get_year_and_measurement(function1) #to make sure the function works
```

```
[1] "Updating long data with year and measurement"

# A tibble: 31,980 x 6
   area_name     STCOU EDU_D      Enrollment  year measurement
   <chr>         <chr> <chr>           <dbl> <dbl> <chr>
 1 UNITED STATES 00000 EDU010197D   44534459  1997 EDU0101
 2 UNITED STATES 00000 EDU010198D   46245814  1998 EDU0101
 3 UNITED STATES 00000 EDU010199D   46368903  1999 EDU0101
 4 UNITED STATES 00000 EDU010200D   46818690  2000 EDU0102
 5 UNITED STATES 00000 EDU010201D   47127066  2001 EDU0102
 6 UNITED STATES 00000 EDU010202D   47606570  2002 EDU0102
 7 UNITED STATES 00000 EDU015203D   48506317  2003 EDU0152
 8 UNITED STATES 00000 EDU015204D   48693287  2004 EDU0152
 9 UNITED STATES 00000 EDU015205D   48978555  2005 EDU0152
10 UNITED STATES 00000 EDU015206D   49140702  2006 EDU0152
# i 31,970 more rows
```

### Function 3: Step 5

Write a function to do Step 5

```r
get_state <- function(county_tibble){
  print("Assigning State to county tibble")
  county_tibble_with_state <- county_tibble |>
  mutate(state = str_trim(str_split(area_name, ",", simplify = TRUE)[,-1]))

  return(county_tibble_with_state)
}
```

### Function 4: Step 6

Write a function to do step 6

```r
get_division <- function(state_tibble){
  print("Assigning division to state tibble")
  state_tibble_with_division <- state_tibble |>
    mutate(division = map_chr(area_name, get_division_for_state))

  return(state_tibble_with_division)
}
```

### Function 5: Step 4

Write another function that takes in the output from Step 3 and creates the two tibbles in Step 4, calls the above two functions (to perform Steps 5 and 6), and returns two final tibbles.

```r
returning_final_tibbles <- function(long_data_updated){
  county_match <- grep(pattern = ", \\w\\w", long_data_updated$area_name)
  county_tibble <- long_data_updated[county_match,]
  class(county_tibble) <- c("county", class(county_tibble))

  state_match <- grep(pattern = ", \\w\\w",
                      long_data_updated$area_name, invert = T)
  state_tibble <- long_data_updated[state_match,]
  class(state_tibble) <- c("state", class(state_tibble))

  county_tibble_final <- get_state(county_tibble)
  state_tibble_final <- get_division(state_tibble)
  return(list(county_tibble_final, state_tibble_final))
}

#making sure the function works
returning_final_tibbles(get_year_and_measurement(function1))
```

```
[1] "Updating long data with year and measurement"
[1] "Assigning State to county tibble"
[1] "Assigning division to state tibble"

[[1]]
# A tibble: 31,450 x 7
   area_name    STCOU EDU_D      Enrollment  year measurement state
   <chr>        <chr> <chr>           <dbl> <dbl> <chr>       <chr>
 1 Autauga, AL  01001 EDU010197D       8099  1997 EDU0101     AL
 2 Autauga, AL  01001 EDU010198D       8211  1998 EDU0101     AL
 3 Autauga, AL  01001 EDU010199D       8489  1999 EDU0101     AL
 4 Autauga, AL  01001 EDU010200D       8912  2000 EDU0102     AL
 5 Autauga, AL  01001 EDU010201D       8626  2001 EDU0102     AL
 6 Autauga, AL  01001 EDU010202D       8762  2002 EDU0102     AL
 7 Autauga, AL  01001 EDU015203D       9105  2003 EDU0152     AL
 8 Autauga, AL  01001 EDU015204D       9200  2004 EDU0152     AL
 9 Autauga, AL  01001 EDU015205D       9559  2005 EDU0152     AL
10 Autauga, AL  01001 EDU015206D       9652  2006 EDU0152     AL
# i 31,440 more rows

[[2]]
# A tibble: 530 x 7
   area_name      STCOU EDU_D      Enrollment  year measurement division
   <chr>          <chr> <chr>           <dbl> <dbl> <chr>       <chr>
 1 UNITED STATES  00000 EDU010197D   44534459  1997 EDU0101     ERROR
 2 UNITED STATES  00000 EDU010198D   46245814  1998 EDU0101     ERROR
 3 UNITED STATES  00000 EDU010199D   46368903  1999 EDU0101     ERROR
 4 UNITED STATES  00000 EDU010200D   46818690  2000 EDU0102     ERROR
 5 UNITED STATES  00000 EDU010201D   47127066  2001 EDU0102     ERROR
 6 UNITED STATES  00000 EDU010202D   47606570  2002 EDU0102     ERROR
 7 UNITED STATES  00000 EDU015203D   48506317  2003 EDU0152     ERROR
 8 UNITED STATES  00000 EDU015204D   48693287  2004 EDU0152     ERROR
 9 UNITED STATES  00000 EDU015205D   48978555  2005 EDU0152     ERROR
10 UNITED STATES  00000 EDU015206D   49140702  2006 EDU0152     ERROR
# i 520 more rows
```

### Wrap Everything in One Function Call (Wrapper Function)

```r
clean_data_wrapper <- function(url, value = "Enrollment"){
  result <- select_and_convert(url, value_colname = value) |>
    get_year_and_measurement() |>
    returning_final_tibbles()

}
```

### Call It and Combine Data

Call the function you made two times to read in and parse the two .csv files mentioned so far. Be sure to call the new value column the same in both function calls.

```r
data_a <- clean_data_wrapper("https://www4.stat.ncsu.edu/~online/datasets/EDU01a.csv")
```

```
[1] "Updating long data with year and measurement"
# A tibble: 5 x 4
  area_name     STCOU EDU_D      Enrollment
  <chr>         <chr> <chr>           <dbl>
1 UNITED STATES 00000 EDU010187D   40024299
2 UNITED STATES 00000 EDU010188D   39967624
3 UNITED STATES 00000 EDU010189D   40317775
4 UNITED STATES 00000 EDU010190D   40737600
5 UNITED STATES 00000 EDU010191D   41385442
[1] "Assigning State to county tibble"
[1] "Assigning division to state tibble"
```

```r
data_b <- clean_data_wrapper("https://www4.stat.ncsu.edu/~online/datasets/EDU01b.csv")
```

```
[1] "Updating long data with year and measurement"
# A tibble: 5 x 4
  area_name     STCOU EDU_D      Enrollment
  <chr>         <chr> <chr>           <dbl>
1 UNITED STATES 00000 EDU010197D   44534459
2 UNITED STATES 00000 EDU010198D   46245814
3 UNITED STATES 00000 EDU010199D   46368903
4 UNITED STATES 00000 EDU010200D   46818690
5 UNITED STATES 00000 EDU010201D   47127066
```

```
[1] "Assigning State to county tibble"
[1] "Assigning division to state tibble"
```

Write a single short function that takes in the results of two calls to your wrapper function. The function should combine the tibbles appropriately (that is the two county level data sets get combined and the two non-county level data sets get combined). This can easily be done within your function using some calls to dplyr::bind_rows(). The function should then return two data sets as one object (in the same format as the input data sets as we will be combining this output with more calls to the wrapper function in a bit).

```
combining_tibbles <- function(tibble1, tibble2){
  county_tibbles_combined <- bind_rows(tibble1[[1]], tibble2[[1]])
  state_tibbles_combined <- bind_rows(tibble1[[2]], tibble2[[2]])
  return(list(county_tibbles_combined, state_tibbles_combined))
}
```

Call this function to combine the result of the two calls to the wrapper function.

```
#saving this test to use it for testing plots later
test_data <- combining_tibbles(data_a, data_b)

test_data #to show that it worked
```

```
[[1]]
# A tibble: 62,900 x 7
   area_name   STCOU EDU_D      Enrollment  year measurement state
   <chr>       <chr> <chr>           <dbl> <dbl> <chr>       <chr>
 1 Autauga, AL 01001 EDU010187D       6829  1987 EDU0101     AL
 2 Autauga, AL 01001 EDU010188D       6900  1988 EDU0101     AL
 3 Autauga, AL 01001 EDU010189D       6920  1989 EDU0101     AL
 4 Autauga, AL 01001 EDU010190D       6847  1990 EDU0101     AL
 5 Autauga, AL 01001 EDU010191D       7008  1991 EDU0101     AL
 6 Autauga, AL 01001 EDU010192D       7137  1992 EDU0101     AL
 7 Autauga, AL 01001 EDU010193D       7152  1993 EDU0101     AL
 8 Autauga, AL 01001 EDU010194D       7381  1994 EDU0101     AL
 9 Autauga, AL 01001 EDU010195D       7568  1995 EDU0101     AL
10 Autauga, AL 01001 EDU010196D       7834  1996 EDU0101     AL
# i 62,890 more rows

[[2]]
# A tibble: 1,060 x 7
```

```
   area_name     STCOU EDU_D      Enrollment  year measurement division
   <chr>         <chr> <chr>           <dbl> <dbl> <chr>       <chr>
 1 UNITED STATES 00000 EDU010187D   40024299  1987 EDU0101     ERROR
 2 UNITED STATES 00000 EDU010188D   39967624  1988 EDU0101     ERROR
 3 UNITED STATES 00000 EDU010189D   40317775  1989 EDU0101     ERROR
 4 UNITED STATES 00000 EDU010190D   40737600  1990 EDU0101     ERROR
 5 UNITED STATES 00000 EDU010191D   41385442  1991 EDU0101     ERROR
 6 UNITED STATES 00000 EDU010192D   42088151  1992 EDU0101     ERROR
 7 UNITED STATES 00000 EDU010193D   42724710  1993 EDU0101     ERROR
 8 UNITED STATES 00000 EDU010194D   43369917  1994 EDU0101     ERROR
 9 UNITED STATES 00000 EDU010195D   43993459  1995 EDU0101     ERROR
10 UNITED STATES 00000 EDU010196D   44715737  1996 EDU0101     ERROR
# i 1,050 more rows
```

## Write Generic Functions for Summarizing

### Plotting State Data

Let’s show commas in our y-axis to make it more readable.

Source: StackOverflow

```r
# We will use the library scales
library(scales)
```

```r
plot.state <- function(df, var_name = "Enrollment"){
  # Create title base on our supplied var name
  plot_title <- paste0("Mean ", var_name, " by Division")

  df |>
    filter(division != "ERROR") |>
    group_by(division, year) |>
    summarize(mean_enrollment = mean(get(var_name))) |>
    mutate(division = as.factor(division)) |>

    # Plotting functions
    ggplot(aes(year,mean_enrollment, color = division)) +
    geom_line() +
    labs(title = plot_title, x = "Year", y = paste0("Mean ", var_name)) +
    guides(color = guide_legend("U.S. Division")) + # Rename Legend
    scale_y_continuous(label=comma)
}
```

### Plotting County Data

```
plot.county <- function(df, var_name = "Enrollment",
                        state = "NC",
                        top_or_bottom = "top", n = 5){
  # Argument validation
  # Try to match by state
  if (is.na(state.name[match(state, state.abb)])){
    stop("Argument Error: state must be two letter state abb, e.g. 'NC' ")
  }
  if (!(all.equal(n, as.integer(n))) == TRUE ){
    stop("Argument Error: Please use an integer for n")
  }


  # Create title based on our supplied var name
  plot_title <- paste0(ifelse(top_or_bottom == "top", "Highest", "Lowest")," ",
                       var_name, " in ", state, " by County")

  # Helper function, not sure if this is the most efficient way to handle this
  display_function <- function(df, col, top_or_bottom){
    if (top_or_bottom == "top"){
      df |> arrange(desc({{col}})) # Nested brackets to refer to the column
    }
    else if(top_or_bottom == "bottom"){
      df |> arrange({{col}})
    }
    else
      stop("Argument Error: top_or_bottom must be 'top' or 'bottom'")
  }

  # Get n top or bottom counties
  counties <- df |>
    filter({{state}} == state ) |>
    group_by(area_name) |>
    summarize(mean_enrollment = mean(get(var_name))) |>
    display_function(mean_enrollment, top_or_bottom) |>
    head(n)
  # Filter df by counties and plot
  # 2 cols for legend so that it's not cut off at the top
  df |> filter(df$area_name %in% counties$area_name) |>
    group_by(year, area_name) |>
    ggplot(aes(year,get(var_name), color = area_name)) + geom_line() +
    labs(title = plot_title, x = "Year", y = paste0(var_name)) +
    guides(color = guide_legend("Location", ncol = 2)) +
    scale_y_continuous(label=comma)


}
```

### Put It Together

Run your data processing function on the two enrollment URLs given previously, specifying an appropriate name for the enrollment data column.

```
data1 <- clean_data_wrapper("https://www4.stat.ncsu.edu/~online/datasets/EDU01a.csv")
```

```
[1] "Updating long data with year and measurement"
# A tibble: 5 x 4
  area_name     STCOU EDU_D      Enrollment
  <chr>         <chr> <chr>           <dbl>
1 UNITED STATES 00000 EDU010187D   40024299
2 UNITED STATES 00000 EDU010188D   39967624
3 UNITED STATES 00000 EDU010189D   40317775
4 UNITED STATES 00000 EDU010190D   40737600
5 UNITED STATES 00000 EDU010191D   41385442
[1] "Assigning State to county tibble"
[1] "Assigning division to state tibble"
```

```
data2 <- clean_data_wrapper("https://www4.stat.ncsu.edu/~online/datasets/EDU01b.csv")
```

```
[1] "Updating long data with year and measurement"
# A tibble: 5 x 4
  area_name     STCOU EDU_D      Enrollment
  <chr>         <chr> <chr>           <dbl>
1 UNITED STATES 00000 EDU010197D   44534459
2 UNITED STATES 00000 EDU010198D   46245814
```

```
3 UNITED STATES 00000 EDU010199D   46368903
4 UNITED STATES 00000 EDU010200D   46818690
5 UNITED STATES 00000 EDU010201D   47127066
[1] "Assigning State to county tibble"
[1] "Assigning division to state tibble"
```

Run your data combining function to put these into one object (with two data frames)

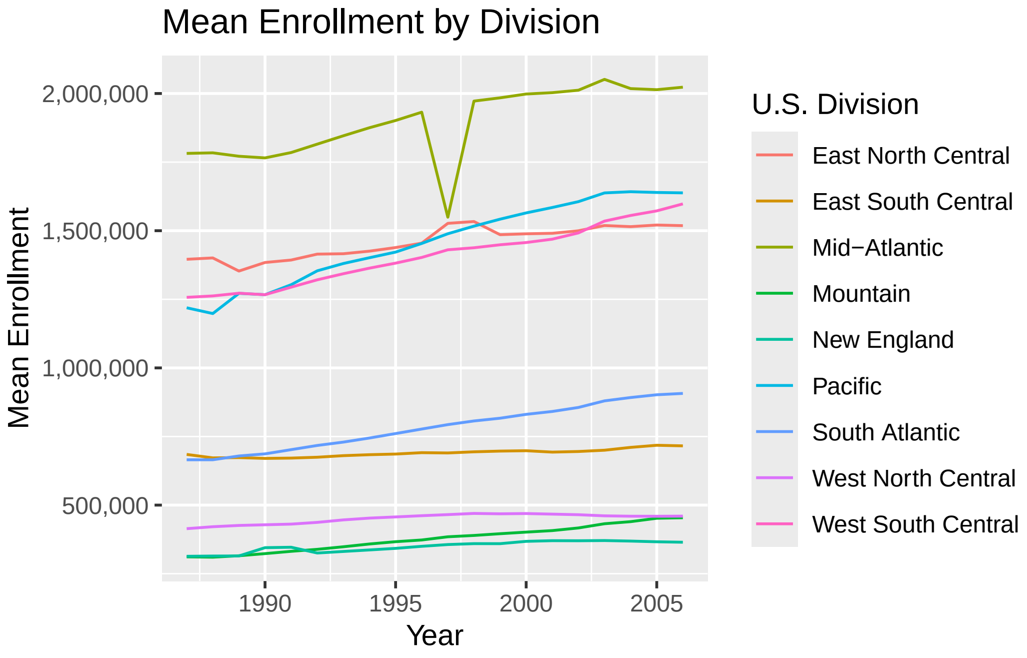
```
one_object <- combining_tibbles(data1, data2)
```

(Use appropriate indexing (ex. [[1]]) to reference the correct data frame)

Use the plot function on the state data frame

```
plot(one_object[[2]])
```

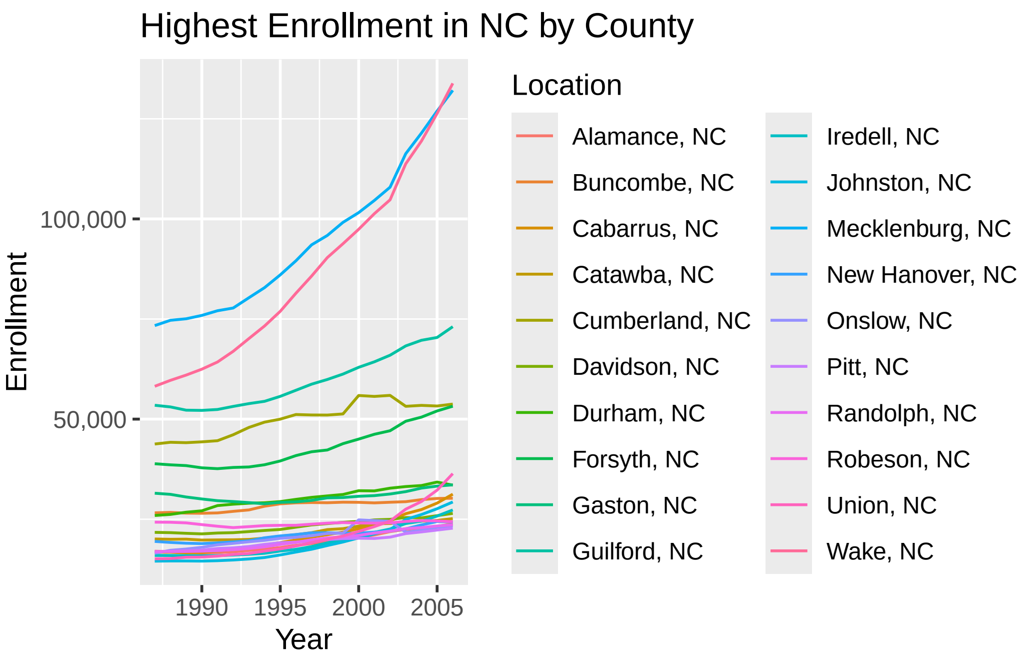
```
`summarise()` has grouped output by 'division'. You can override using the
`.groups` argument.
```



Use the plot function on the county data frame

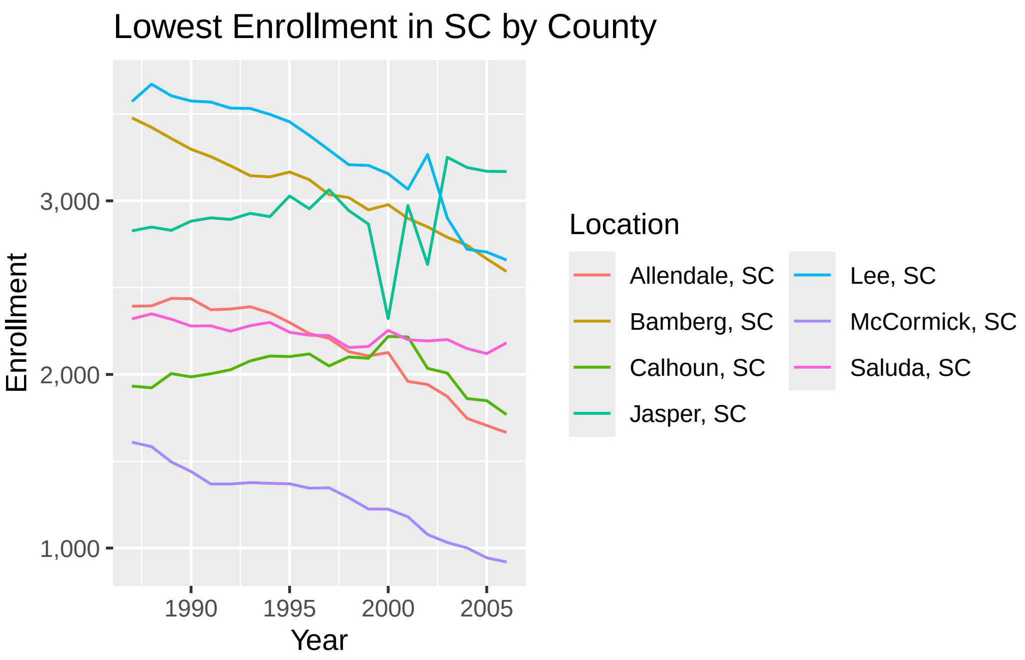
– Once specifying the state to be "NC", the group being the top, the number looked at being 20

```
plot(one_object[[1]], top_or_bottom = "top", n = 20, state = "NC")
```



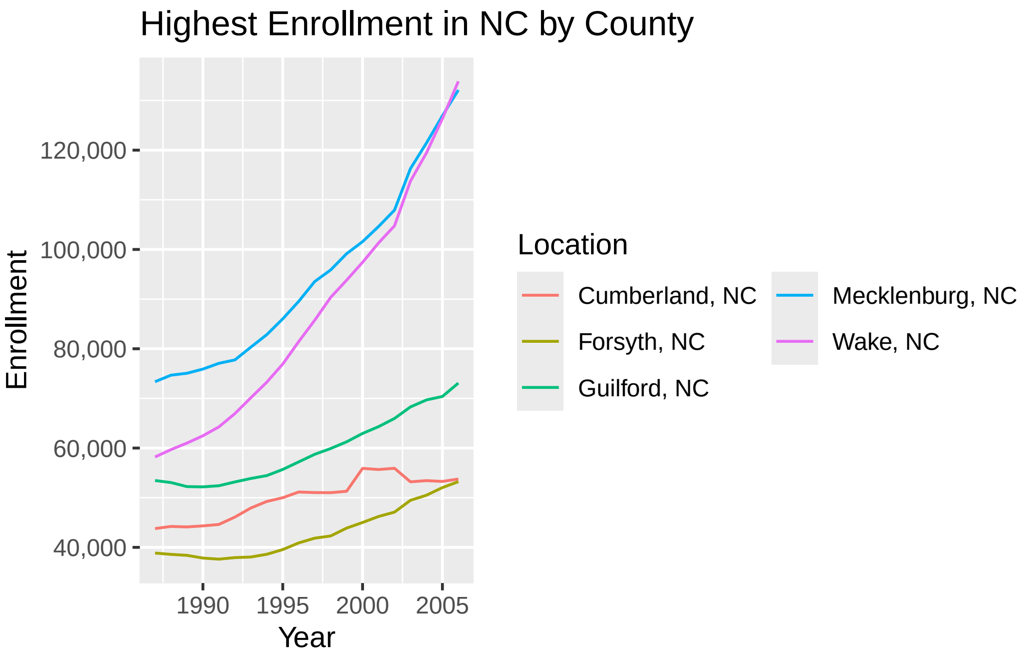
– Once specifying the state to be “SC”, the group being the bottom, the number looked at being 7

```
plot(one_object[[1]], top_or_bottom = "bottom", n = 7, state = "SC")
```



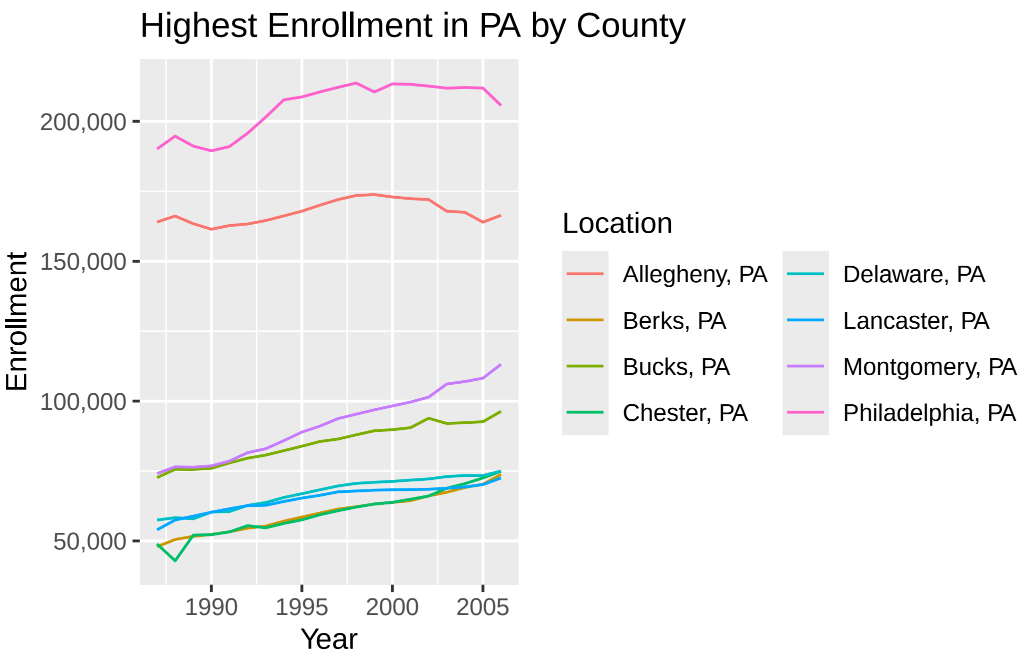
– Once without specifying anything (defaults used)

```
plot(one_object[[1]])
```



– Once specifying the state to be “PA”, the group being the top, the number looked at being 8

```
plot(one_object[[1]], top_or_bottom = "top", n = 8, state = "PA")
```



Lastly, read in another couple similar data sets and apply your functions!

Run your data processing function on the four data sets at URLs given.

After referring to the Mastdata spreadsheet, We must note that the data below corresponds to “Resident Total Population” estimates, rather than enrollment. So, we will pass the approprriate value to our wrapper functions.

```
dataPa <- clean_data_wrapper("https://www4.stat.ncsu.edu/~online/datasets/PST01a.csv",
                             value = "Resident Population")
```

```
[1] "Updating long data with year and measurement"
# A tibble: 5 x 4
  area_name     STCOU EDU_D      `Resident Population`
  <chr>         <chr> <chr>                      <dbl>
1 UNITED STATES 00000 PST015171D             206827028
2 UNITED STATES 00000 PST015172D             209283904
3 UNITED STATES 00000 PST015173D             211357490
```

```
4 UNITED STATES 00000 PST015174D             213341552
5 UNITED STATES 00000 PST015175D             215465246
[1] "Assigning State to county tibble"
[1] "Assigning division to state tibble"
```

```
dataPb <- clean_data_wrapper("https://www4.stat.ncsu.edu/~online/datasets/PST01b.csv",
                             value = "Resident Population")
```

```
[1] "Updating long data with year and measurement"
# A tibble: 5 x 4
  area_name     STCOU EDU_D      `Resident Population`
  <chr>         <chr> <chr>                      <dbl>
1 UNITED STATES 00000 PST025182D             231665106
2 UNITED STATES 00000 PST025183D             233792697
3 UNITED STATES 00000 PST025184D             235825544
4 UNITED STATES 00000 PST025185D             237924311
5 UNITED STATES 00000 PST025186D             240133472
[1] "Assigning State to county tibble"
[1] "Assigning division to state tibble"
```

```
dataPc <- clean_data_wrapper("https://www4.stat.ncsu.edu/~online/datasets/PST01c.csv",
                             value = "Resident Population")
```

```
[1] "Updating long data with year and measurement"
# A tibble: 5 x 4
  area_name     STCOU EDU_D      `Resident Population`
  <chr>         <chr> <chr>                      <dbl>
1 UNITED STATES 00000 PST035191D             252980941
2 UNITED STATES 00000 PST035192D             256514224
3 UNITED STATES 00000 PST035193D             259918588
4 UNITED STATES 00000 PST035194D             263125821
5 UNITED STATES 00000 PST035195D             266278393
[1] "Assigning State to county tibble"
[1] "Assigning division to state tibble"
```

```
dataPd <- clean_data_wrapper("https://www4.stat.ncsu.edu/~online/datasets/PST01d.csv",
                             value = "Resident Population")
```

```
[1] "Updating long data with year and measurement"
# A tibble: 5 x 4
```

```
  area_name     STCOU EDU_D      `Resident Population`
  <chr>         <chr> <chr>                      <dbl>
1 UNITED STATES 00000 PST045200D             282171957
2 UNITED STATES 00000 PST045201D             285081556
3 UNITED STATES 00000 PST045202D             287803914
4 UNITED STATES 00000 PST045203D             290326418
5 UNITED STATES 00000 PST045204D             293045739
[1] "Assigning State to county tibble"
[1] "Assigning division to state tibble"
```

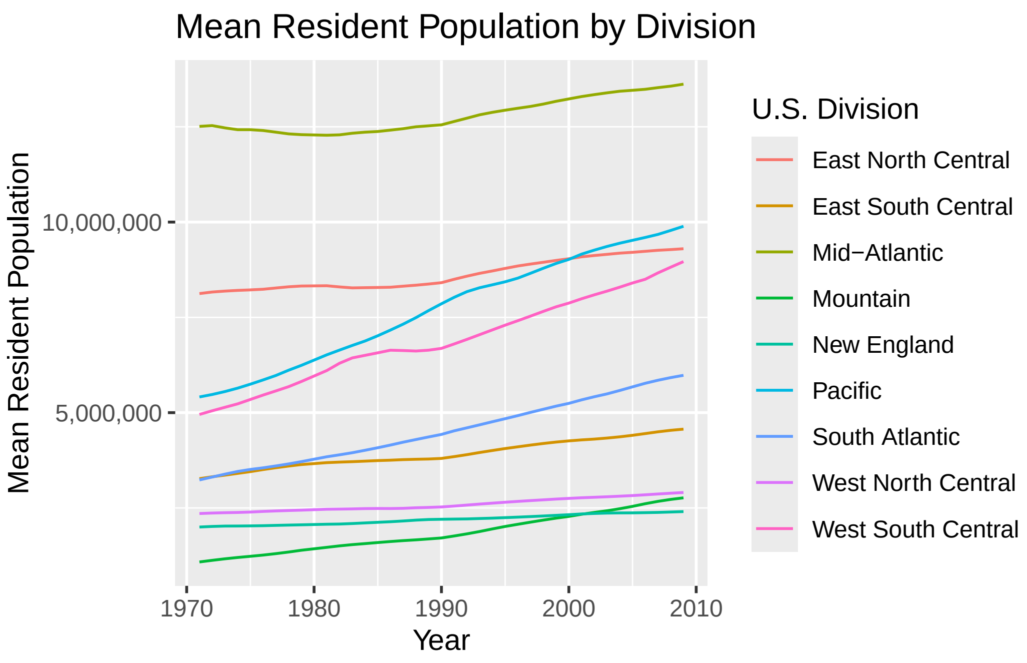
Run your data combining function (probably three times) to put these into one object (with two data frames)

```
once <- combining_tibbles(dataPa, dataPb)

twice <- combining_tibbles(once, dataPc)

thrice <- combining_tibbles(twice, dataPd)
```

Use the plot function on the state data frame

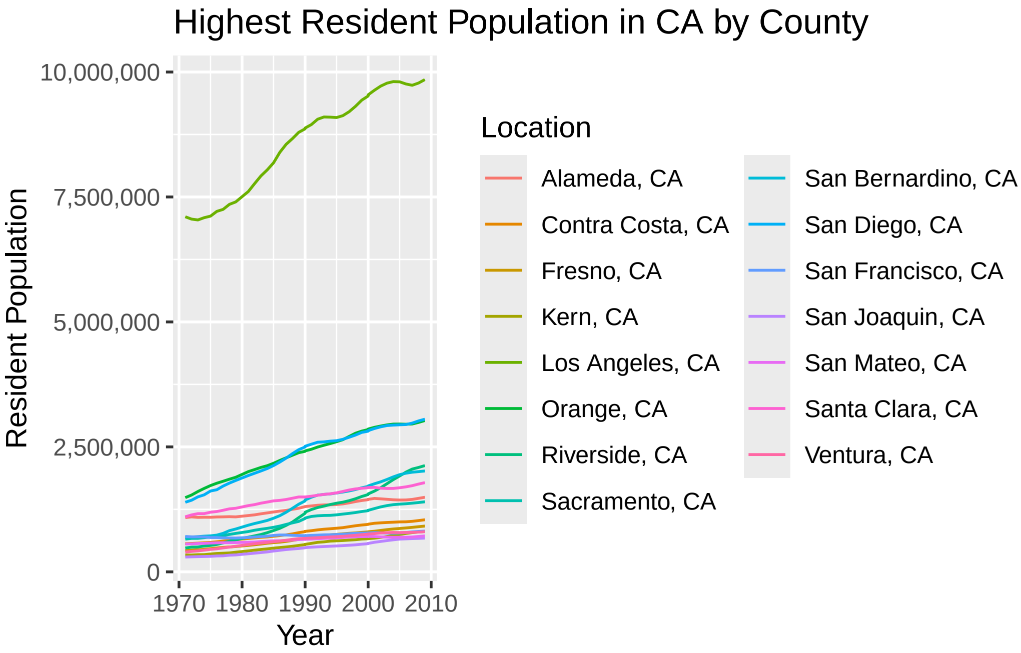
```
plot(thrice[[2]], var_name = "Resident Population")
```

```
`summarise()` has grouped output by 'division'. You can override using the
`.groups` argument.
```



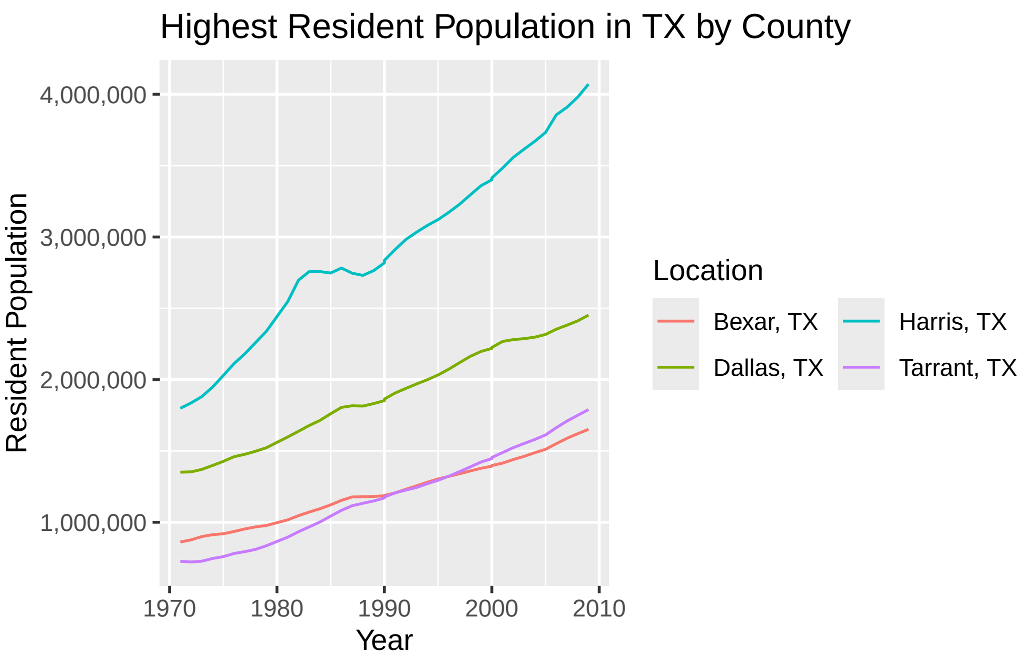
Use the plot function on the county data frame – Once specifying the state to be "CA", the group being the top, the number looked at being 15

```r
plot(thrice[[1]], top_or_bottom = "top", n = 15, state = "CA",
     var_name = "Resident Population")
```



– Once specifying the state to be "TX", the group being the top, the number looked at being 4

```
plot(thrice[[1]], top_or_bottom = "top", n = 4, state = "TX",
     var_name = "Resident Population")
```



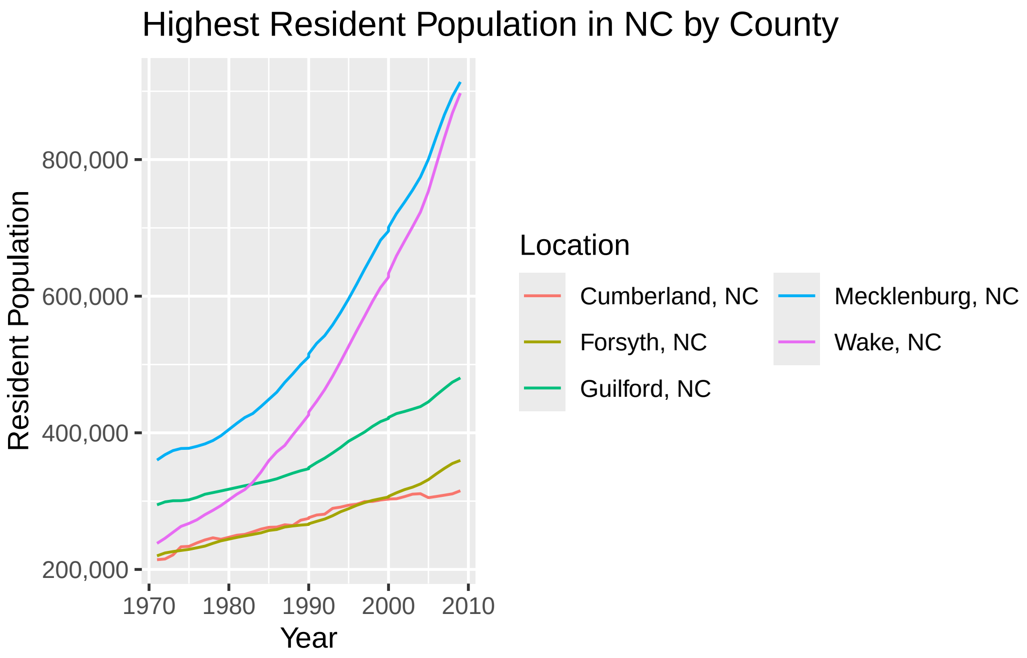
– Once without specifying anything (defaults used)

```
# Note: Using the default of "Enrollment" will throw an error since our wrapper
#function was called "Resident Population"
# We will run it here to show the error, and then once with the correct values
tryCatch(
  {plot(thrice[[1]])},
  error = function(e) {
    print(e)
  }
)
```

```
<error/rlang_error>
Error in `summarize()`:
i In argument: `mean_enrollment = mean(get(var_name))`.
i In group 1: `area_name = "Alamance, NC"`.
Caused by error in `get()`:
! object 'Enrollment' not found
---
Backtrace:
     x
  1. +-base::tryCatch(...)
  2. | \-base (local) tryCatchList(expr, classes, parentenv, handlers)
```

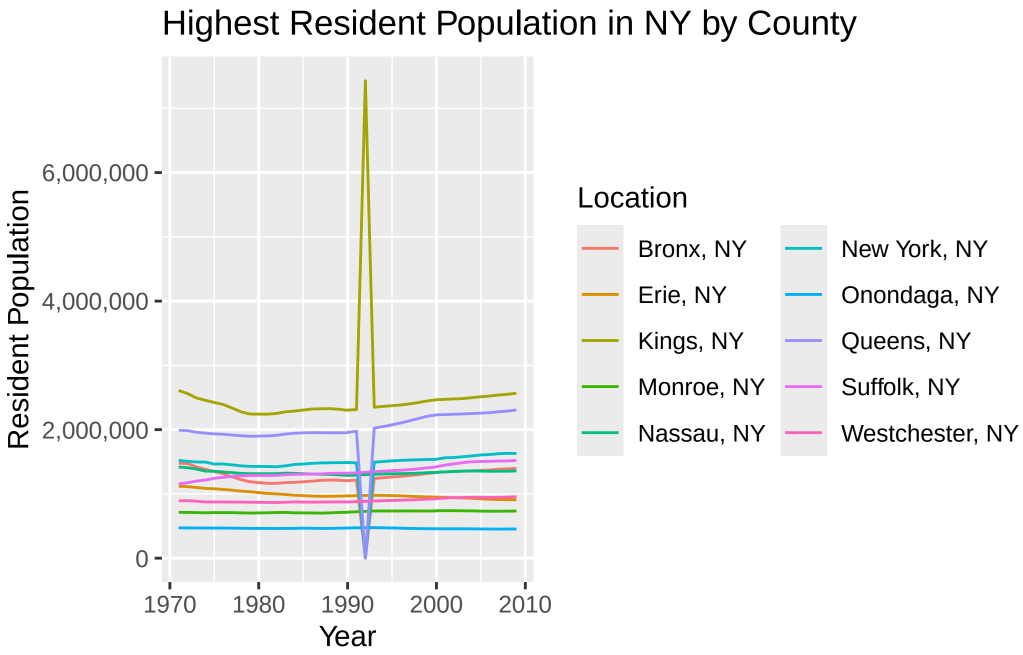
```
 3. |   \-base (local) tryCatchOne(expr, names, parentenv, handlers[[1L]])
 4. |     \-base (local) doTryCatch(return(expr), name, parentenv, handler)
 5. +-base::plot(thrice[[1]])
 6. +-global plot.county(thrice[[1]])
 7. | +-utils::head(...)
 8. | +-display_function(...)
 9. | | \-dplyr::arrange(...)
10. | +-dplyr::summarize(...)
11. | \-dplyr:::summarise.grouped_df(...)
12. |   \-dplyr:::summarise_cols(.data, dplyr_quosures(...), by, "summarise")
13. |     +-base::withCallingHandlers(...)
14. |     \-dplyr:::map(quosures, summarise_eval_one, mask = mask)
15. |       \-base::lapply(.x, .f, ...)
16. |         \-dplyr (local) FUN(X[[i]], ...)
17. |           \-mask$eval_all_summarise(quo)
18. |             \-dplyr (local) eval()
19. +-base::mean(get(var_name))
20. \-base::get(var_name)
```

```r
plot(thrice[[1]], var_name = "Resident Population")
```



– Once specifying the state to be “NY”, the group being the top, the number looked at being 10

```
plot(thrice[[1]], top_or_bottom = "top", n = 10, state = "NY",
     var_name = "Resident Population")
```



One thing to note in the graph above are the large spikes/drops in the 1990s - this would warrant further investigation and is a perfect example why EDA is important.